PATENT COOPERATION TREATY

From the INTERNATIONAL BUREAU

PCT

NOTIFICATION OF ELECTION

(PCT Rule 61.2)

To:

Assistant Commissioner for Patents
United States Patent and Trademark Office
Box PCT
Washington, D.C.20231
ETATS-UNIS D'AMERIQUE

in its capacity as elected Office

Date of mailing (day/month/year) 22 September 2000 (22.09.00)	
International application No. PCT/SG99/00009	**Applicant's or agent's file reference** FP1122
International filing date (day/month/year) 22 February 1999 (22.02.99)	**Priority date** (day/month/year) 16 December 1998 (16.12.98)
Applicant NGAIR, Teow, Hin et al	

1. The designated Office is hereby notified of its election made:

[X] in the demand filed with the International Preliminary Examining Authority on:

30 June 2000 (30.06.00)

[ ] in a notice effecting later election filed with the International Bureau on:

2. The election [X] was

[ ] was not

made before the expiration of 19 months from the priority date or, where Rule 32 applies, within the time limit under Rule 32.2(b).

The International Bureau of WIPO
34, chemin des Colombettes
1211 Geneva 20, Switzerland

Facsimile No.: (41-22) 740.14.35

Authorized officer

Lazar Joseph Panakal

Telephone No.: (41-22) 338.83.38

09185411

Copy for the Elected Office (EO/US)

PCT/SG99/00009

PATENT COOPERATION TREATY

From the INTERNATIONAL BUREAU

PCT

NOTIFICATION OF THE RECORDING OF A CHANGE

(PCT Rule 92bis.1 and Administrative Instructions, Section 422)

To:

GREENE-KELLY, James, Patrick
Lloyd Wise
Tanjong Pagar
P.O. Box 636
Singapore 910816
SINGAPOUR

Date of mailing (day/month/year)
13 August 2001 (13.08.01)

Applicant's or agent's file reference
FP1122

IMPORTANT NOTIFICATION

International application No.
PCT/SG99/00009

International filing date (day/month/year)
22 February 1999 (22.02.99)

1. The following indications appeared on record concerning:

[X] the applicant [X] the inventor [ ] the agent [ ] the common representative

Name and Address
PANG, Hwee, Hwa
19 Shelford Road #01-42
Singapore 288408
Singapore

State of Nationality: SG
State of Residence: SG
Telephone No.:
Facsimile No.:
Teleprinter No.:

2. The International Bureau hereby notifies the applicant that the following change has been recorded concerning:

[ ] the person [ ] the name [X] the address [ ] the nationality [ ] the residence

Name and Address
PANG, Hwee, Hwa
201 Tanjong Rhu Road #15-11
Singapore 439617
Singapore

State of Nationality: SG
State of Residence: SG
Telephone No.:
Facsimile No.:
Teleprinter No.:

3. Further observations, if necessary:

4. A copy of this notification has been sent to:

[X] the receiving Office
[ ] the International Searching Authority
[X] the International Preliminary Examining Authority
[ ] the designated Offices concerned
[X] the elected Offices concerned
[ ] other:

The International Bureau of WIPO
34, chemin des Colombettes
1211 Geneva 20, Switzerland

Facsimile No.: (41-22) 740.14.35

Authorized officer
Maria Victoria CORTIELLO

Telephone No.: (41-22) 338.83.38

Form PCT/IB/306 (March 1994)

004208324

PATENT COOPERATION TREATY

PCT

INTERNATIONAL PRELIMINARY EXAMINATION REPORT

(PCT Article 36 and Rule 70)

Applicant's or agent's file reference	FOR FURTHER ACTION	See Notification of Transmittal of International Preliminary Examination Report (Form PCT/IPEA/416)
FP1122		
International application No.	International filing date (*day/month/year*)	Priority Date (*day/month/year*)
PCT/SG 99/00009	22 February 1999 (22.02.1999)	16 December 1998 (16.12.1998)

International Patent Classification (IPC) or national classification and IPC

IPC[7]: G06F 9/46, 9/54

Applicant

Kent Ridge Digital Labs

1. This international preliminary examination report has been prepared by this International Preliminary Examination Authority and is transmitted to the applicant according to Article 36.

2. This REPORT consists of a total of 5 sheets, including this cover sheet.

☐ This report is also accompanied by ANNEXES, i.e., sheets of the description, claims and/or drawings which have been amended and are the basis for this report and/or sheets containing rectifications made before this Authority (see Rule 70.16 and Section 607 of the Administrative Instructions under the PCT).

These annexes consist of a total of _____ sheets.

3. This report contains indications relating to the following items:

- I. ☒ Basis of the opinion
- II. ☐ Priority
- III. ☐ Non-establishment of opinion with regard to novelty, inventive step and industrial applicability
- IV. ☐ Lack of unity of invention
- V. ☒ Reasoned statement under Rule 66.2(a)(ii) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
- VI. ☐ Certain documents cited
- VII. ☐ Certain defects in the international application
- VIII. ☐ Certain observations on the international application

Date of submission of the demand	Date of completion of this report
30.06.2000	24 June 2003 (24.06.2003)
Name and mailing address of the IPEA/AT Austrian Patent Office Dresdner Straße 87 A-1200 Vienna Facsimile No. 1/53424/200	Authorized officer FASTENBAUER K. Telephone No. 1/53424/447

INTERNATIONAL PRELIMINARY EXAMINATION REPORT

International application No.
PCT/SG 99/00009

I. Basis of the report

1. With regard to the **elements** of the international application:*

☒ the international application as originally filed

☐ the description:
pages _____, as originally filed
pages _____, filed with the demand
pages _____, filed with the letter of _____.

☐ the claims:
pages _____, as originally filed
pages _____, as amended (together with any statement) under Article 19
pages _____, filed with the demand
pages _____, filed with the letter of _____.

☐ the drawings:
pages _____, as originally filed
pages _____, filed with the demand
pages _____, filed with the letter of _____.

☐ the sequence listing part of the description:
pages _____, as originally filed
pages _____, filed with the demand
pages _____, filed with the letter of _____.

2. With regard to the **language**, all the elements marked above were available or furnished to this Authority in the language in which the international application was filed, unless otherwise indicated under this item.
These elements were available or furnished to this Authority in the following language _____ which is:

☐ the language of a translation furnished for the purposes of international search (under Rule 23.1(b)).

☐ the language of publication of the international application (under Rule 48.3(b)).

☐ the language of the translation furnished for the purposes of international preliminary examination (under Rule 55.2 and/or 55.3).

3. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, the international preliminary examination was carried out on the basis of the sequence listing:

☐ contained in the international application in printed form.

☐ filed together with the international application in computer readable form.

☐ furnished subsequently to this Authority in written form.

☐ furnished subsequently to this Authority in computer readable form.

☐ The statement that the subsequently furnished written sequence listing does not go beyond the disclosure in the international application as filed has been furnished.

☐ The statement that the information recorded in computer readable form is identical to the written sequence listing has been furnished.

4. ☐ The amendments have resulted in the cancellation of:

☐ the description, pages _____.

☐ the claims, Nos. _____.

☐ the drawings, sheets/fig _____.

5. ☐ This report has been established as if (some of) the amendments had not been made, since they have been considered to go beyond the disclosure as filed, as indicated in the Supplemental Box (Rule 70.2(c)).**

* *Replacement sheets which have been furnished to the receiving Office in response to an invitation under Article 14 are referred to in this report as „originally filed" and are not annexed to this report since they do not contain amendments (Rules 70.16 and 70.17).*

** *Any replacement sheet containing such amendments must be referred to under item 1 and annexed to this report.*

V. Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement

1. Statement

Novelty (N)	Claims	1-22	YES
	Claims		NO
Inventive step (IS)	Claims	1-22	YES
	Claims		NO
Industrial applicability (IA)	Claims	1-22	YES
	Claims		NO

Citations and explanations (Rule 70.7)

The following documents are cited in the search report:

D1) US 5 603 031 A (HELGERSON et al.), 11. Feb. 1997
*** Summary of the invention, para. 4,8,23 relative to claims 1-3,6
totality relative to claims 4-5, 7-22

D2) openUTM V4.0 (BS2000/OSD), Concepts and functions,
Chapter 5: Structure of UTM Applications
5.2: The process concept

D3) BS2000/OSD-BC V1.0, Performance Handbook, Chapter 5.3.1: Managing the resource main memory

D4) WO 97/35262 A (HITACHI), 25. September 1997

Document D1, US 5 603 031, discloses a system and method for distributed computation based upon the movement, execution and interaction in a network, in which agent processes direct their own movement through the computer network. Place processes provide a computing context in which agent processes are interpreted ... An agent process which moves from one place process to another transports definitions of classes of which objects included in the agent process are members. An agent process which moves from one place to a second place process avoids unnecessary transportation of objects included in the agent process by substituting equivalent objects which are found in the second place process.

To transport a particular process, the process is suspended and the execution state is preserved. The instructions of the process, the preserved execution state and objects owned by the process are packaged or "encoded" (§4 of the summary of the invention).

Supplemental Box
(To be used when the space in any of the preceding boxes is not sufficient)

Continuation of: Box V (page 1)

The power of agents is counterbalanced by "permits". A permit limits the particular capabilities of a particular agent on particular occasions (§8 of the summary of the invention).

An agent, occupying a first place, is also capable of creating one or more clones of the agent and moving each clone to a respective place. A clone is an agent process which is the result of duplicating an existing agent process (§23 of the summary of the invention).

Document D2, openUTM Concepts and functions, discloses the concepts and functions of openUTM. The purpose of an UTM application is to provide services: it processes the service requests (jobs) received from terminal users, client programs or other applications. An UTM application consists of an UTM application program which is started in a certain number of processes defined by the user, the KDCFILE which is used by all processes as the "system memory", and a UTM cache memory which optimises access to the KDCFILE. From the technical point of view, an UTM application is a group of processes which form a logical server unit at runtime (Chapter 5: Structure of a UTM application).

In 5.1 "UTM application program" it is disclosed that service routines (or program units) access the UTM system functions through integrated UTM calls. The service routines are assigned transactions codes (TACs) which are used by the terminal users, clients or other programs to start the service routines. To ensure that the service routines can run under the management of openUTM, the compiled service routines are linked to the UTM application program, together with other modules (allocation table, messages, libraries used etc.)
This linking results in a part of the application program known as the **main routine** KDCROOT, which acts as the main control program and is responsible for coordinating job sequences.

In 5.2 "The process concept" it is disclosed that when the application is started, the application program is initiated in a certain number of processes defined by the user. Since a UTM application is generally accessed simultaneously by a number of clients, each client does not have its own exclusive process. Instead, the large number of simultaneous requests is distributed by openUTM across a small number of processes. As a result, the system overhead and thus the response times do not increase proportionally with the number of users.

Furthermore, if the number of free processes exceeds the number of outstanding jobs, the free processes are placed in a process queue and re-activated when needed.

Document D3, BS2000/OSD-BC V1.0, Performance Handbook, discloses in his chapter 5.3.1, Managing the resource main memory, that the following concepts are employed:

- activation
- deactivation
- forced deactivation
- pre-emption
- control functions of main memory management
- page management algorithms

Supplemental Box
(To be used when the space in any of the preceding boxes is not sufficient)

Continuation of: Box V (page 2)

Deactivation of active tasks makes main memory available for inactive, ready tasks.

Document D4, WO 97/35262 , discloses an information system in which a plurality of computers are connected to each other. An information table in which information called "passport" including bibliographic items, such as the destinations, purposes, times and dates of visits, departure times and dates etc. and required to move a program is incorporated in the program. The handling of the program itself while the program visits a server is determined based on the passport information. The program has electronic money and uses the money as the change for the use of a resource on the server. Depending on the processing, a child program which is a copy of the program is generated so as to shorten the processing time.

The examiner agrees with the arguments filed in the letter of July 10th. 2001, that none of the documents shows a method for **removing a part of a computing process**, but only a method for **duplicating an existing process.**

Even though D1 discloses the possibility of not including all the information of the agent in the clones, but only those parts which are not needed in the target-system, D1 does not disclose a method of splitting the process in a first and a second sub-process comprising the parts not required by the first process.

Also knowing D2 and D3, disclosing a method of placing existing sub-processes not needed at the moment in a queue and storing them subsequently in a memory storage device, does not lead a person skilled in the art to the ides of creating a new sub-process which will be sent to a memory storage device.

Therefore all of the claims 1-22 may be considered as being new and involving an inventive step.

The industrial applicability is given for all claims.

PATENT COOPERATION TREATY

PCT

INTERNATIONAL SEARCH REPORT

(PCT Article 18 and Rules 43 and 44)

Applicant's or agent's file reference	FOR FURTHER ACTION	see Notification of Transmittal of International Search Report (Form PCT/ISA/220) as well as, where applicable, item 5 below.
FP1122		
International application No.	International filing date (*day/month/year*)	(Earliest) Priority Date (*day/month/year*)
PCT/SG 99/00009	22 February 1999 (22.02.99)	16. December 1998 (16.12.98)
Applicant		
KENT RIDGE DIGITAL LABS et al.		

This international search report has been prepared by this International Searching Authority and is transmitted to the applicant according to Article 18. A copy is being transmitted to the International Bureau.

This international search report consists of a total of 3 sheets.

☐ It is also accompanied by a copy of each prior art document cited in this report.

1. **Basis of the report**
   a. With regard to the **language**, the international search was carried out on the basis of the international application in the language in which it was filed, unless otherwise indicated under this item.
      ☐ the international search was carried out on the basis of a translation of the international application furnished to this Authority (Rule 23.1(b)).
   b. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, the international search was carried out on the basis of the sequence listing:
      ☐ contained in the international application in written form.
      ☐ filed together with the international application in computer readable form.
      ☐ furnished subsequently to this Authority in written form.
      ☐ furnished subsequently to this Authority in computer readable form.
      ☐ the statement that the subsequently furnished written sequence listing does not go beyond the disclosure in the international application as filed has been furnished.
      ☐ the statement that the information recorded in computer readable form is identical to the written sequence listing has been furnished.
2. ☐ **Certain claims were found unsearchable** (See Box I).
3. ☐ **Unity of invention is lacking** (See Box II).
4. With regard to the **title**,
   ☒ the text is approved as submitted by the applicant.
   ☐ the text has been established by this Authority to read as follows:
5. With regard to the **abstract**,
   ☒ the text is approved as submitted by the applicant.
   ☐ the text has been established, according to Rule 38.2(b), by this Authority as it appears in Box III. The applicant may, within one month from the date of mailing of this international search report, submit comments to this Authority.
6. The figure of the **drawings** to be published with the abstract is Figure No.: 2
   ☒ as suggested by the applicant. ☐ None of the figures.
   ☐ because the applicant failed to suggest a figure.
   ☐ because this figure better characterizes the invention.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SG 99/00009

A. CLASSIFICATION OF SUBJECT MATTER

IPC[7]: G06F 9/46, 9/54

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC[7]: G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPODOC, WPI, The INTERNET

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X A	US 5603031A (HELGESON et al.), 11 February 1997 (11.02.97). Summary of the Invention, paragraph 4,8,23 totality.	1-3,6 4-5, 7-22
A	openUTM V4.0 (BS2000/OSD), Concepts and Functions, Chapter 5: Structure of UTM Applications, especially 5.2: The process concept, paragraph 3, page 67, ID: U20683-J-Z135-2-7600 [online] February 1997 [retrieved on 1999-12-13]. Retrieved from the Internet via <URL: http//manuals.siemens.de/servers/bs2_man/man_us/utm/v4_0/utm_kon.pdf>.	7-10
A	BS2000/OSD-BC V1.0, Performance Handbook, Chapter 5.3.1: Managing the resource main memory, especially: Deactivation (p.248), Waiting Time runout Control (p.255), Paging management algorithms (pp.256-257), ID: U1794-J-Z125-6-7600 [online] February 1997 [retrieved on 1999-12-13]. Retrieved from the Internet via <URL:http://manuals.siemens.de/servers/bs2_man/man_us/bs2_bc/V1_0/perform.pdf>.	7-10
A	WO 9735262 A (HITACHI), 25 September 1997 (25.09.97).	1-22

☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:
„A" document defining the general state of the art which is not considered to be of particular relevance
„E" earlier application or patent but published on or after the international filing date
„L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
„O" document referring to an oral disclosure, use, exhibition or other means
„P" document published prior to the international filing date but later than the priority date claimed
„T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
„X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
„Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
„&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
14 December 1999 (14.12.99)	14 January 2000 (14.01.00)

Name and mailing adress of the ISA/AT	Authorized officer
Austrian Patent Office Kohlmarkt 8-10; A-1014 Vienna Facsimile No. 1/53424/200	Fastenbauer Telephone No. 1/53424/447

Form PCT/ISA/210 (second sheet) (July 1998)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/SG 99/00009

Patent document cited in search report			Publication date	Patent family member(s)			Publication date
US	A	5603031	11-02-1997	AU	A1	72164/94	06-02-1995
				EP	A2	634719	18-01-1995
				EP	A3	634719	03-01-1996
				JP	A2	7182174	21-07-1995
				JP	T2	7509799	26-10-1995
				WO	A1	9502219	19-01-1995
WO	A1	9735262	25-09-1997			none	

PATENT COOPERATION TREATY

PCT

INTERNATIONAL SEARCH REPORT

(PCT Article 18 and Rules 43 and 44)

Applicant's or agent's file reference	FOR FURTHER ACTION	see Notification of Transmittal of International Search Report (Form PCT/ISA/220) as well as, where applicable, item 5 below.
FP 1123		
International application No.	International filing date (*day/month/year*)	(Earliest) Priority Date (*day/month/year*)
PCT/SG 99/00018	18 March 1999 (18.03.99)	16 December 1998 (16.12.98)
Applicant		
KENT RIDGE DIGITAL LABS et al.		

This international search report has been prepared by this International Searching Authority and is transmitted to the applicant according to Article 18. A copy is being transmitted to the International Bureau.

This international search report consists of a total of 6 sheets.

☐ It is also accompanied by a copy of each prior art document cited in this report.

1. **Basis of the report**
   a. With regard to the **language**, the international search was carried out on the basis of the international application in the language in which it was filed, unless otherwise indicated under this item.
      ☐ the international search was carried out on the basis of a translation of the international application furnished to this Authority (Rule 23.1(b)).
   b. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, the international search was carried out on the basis of the sequence listing:
      ☐ contained in the international application in written form.
      ☐ filed together with the international application in computer readable form.
      ☐ furnished subsequently to this Authority in written form.
      ☐ furnished subsequently to this Authority in computer readable form.
      ☐ the statement that the subsequently furnished written sequence listing does not go beyond the disclosure in the international application as filed has been furnished.
      ☐ the statement that the information recorded in computer readable form is identical to the written sequence listing has been furnished.
2. ☐ **Certain claims were found unsearchable** (See Box I).
3. ☒ **Unity of invention is lacking** (See Box II).
4. With regard to the **title**,
   ☒ the text is approved as submitted by the applicant.
   ☐ the text has been established by this Authority to read as follows:
5. With regard to the **abstract**,
   ☒ the text is approved as submitted by the applicant.
   ☐ the text has been established, according to Rule 38.2(b), by this Authority as it appears in Box III. The applicant may, within one month from the date of mailing of this international search report, submit comments to this Authority.
6. The figure of the **drawings** to be published with the abstract is Figure No.: 2
   ☒ as suggested by the applicant. ☐ None of the figures.
   ☐ because the applicant failed to suggest a figure.
   ☐ because this figure better characterizes the invention.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SG 99/00018

Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

cl. 1 ... a method for migrating a computing process ... discards data
cl. 2 ... ... discards data ... prior to migration
cl. 3-4 ... ... a construct is formed ... / suspend all active threads
cl. 5 ... ... falling whithin lists
cl. 6 ... ... authorizing signature
cl. 7 ... ... sent directly to said second host
cl. 8 ... ... sent to an intermediate storage means ...
cl. 9 ... ... second process is run on said second host ...
cl. 10 ... ... third process ...

1. ☒ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest
☐ The additional search fees were accompanied by the applicant's protest.
☐ No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SG 99/00018

continuation of first sheet (1):

cl. 11-17	... discards data ... after migration
cl. 21-30	... discards data ... after migration ... and after receiving data ...
cl. 31	... relates to library modules and/or input/output drivers of said first host
cl. 32	... relates to library modules and/or input/output drivers of said second host

INTERNATIONAL SEARCH REPORT

International application No.
PCT / SG 99/00018

A. CLASSIFICATION OF SUBJECT MATTER

IPC[6]: G06F 9/46, G06F 9/54

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC[6]: G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPODOC, WPI, The INTERNET, ACM Digital Library

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
✓X A	US 5603031A (HELGESON et al.) 11 February 1997 (11.2.1997), summary of the invention, col. 7 li. 66 - col.8 li. 10, col. 8 li. 51 ff., col. 10 li. 31-col. 11 li. 28 , fig. 24A-C, totality.	1-10, 31,32 11-30
✓X A	ARIDOR, Yariv, and LANGE Danny B. Agent design patterns: elements of agent application design. Proceedings of the second international conference of Autonomous agents, Minneapolis, MN USA, p.108-115 [online], May 10-13,1998 [retrieved on 2000-05-17]. Retrieved from the Internet URL:< http://www.acm.org/pubs/articles/proceedings/ai /280765/p108-aridor/p108-aridor.pdf>.	1-3,7,9,31,32 4-6,8,10-30
✓A	CHEN Qiming, CHUNDI Parvathi, DAYAL Umeshwar, and HSU Meichun. Dynamic software agents for business intelligence applications (poster). Proceedings of the second international conference of Autonomous agents, Minneapolis, MN USA, p.453-454 [online], May 10-13,1998 [retrieved on 2000-05-17]. Retrieved from the Internet URL:<http://www.acm.org/pubs/articles/ proceedings/ai/280765/p453-chen/p453-chen.pdf>.	1-32

☒ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:
„A" document defining the general state of the art which is not considered to be of particular relevance
„E" earlier application or patent but published on or after the international filing date
„L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
„O" document referring to an oral disclosure, use, exhibition or other means
„P" document published prior to the international filing date but later than the priority date claimed
„T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
„X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
„Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
„&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
18 May 2000 (18.05.2000)	19 May 2000 (19.05.00)
Name and mailing adress of the ISA/AT Austrian Patent Office Kohlmarkt 8-10; A-1014 Vienna Facsimile No. 1/53424/535	Authorized officer Fastenbauer Telephone No. 1/53424/447

Form PCT/ISA/210 (second sheet) (July 1998)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SG99/00018

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CHAUHAN Deepika and BAKER Albert D. JAFMAS: a multiagent application development system. Proceedings of the second international conference of Autonomous agents, Minneapolis, MN USA, p. 100-107 [online], May 10-13,1998 [retrieved on 2000-05-17]. Retrieved from the Internet URL:< http://www.acm.org/pubs/citations/proceedings /ai/280765/p100-chauhan/ p100-chauhan.pdf>.	1-32
A	PKZIP (R) FAST! Create/Update Utility Version 2.04g, 1993-01-02. Copyright 1989-1993 PKWARE Inc. Shareware Version.	1-32
A	WO 97/35262 A (HITACHI), 25 September 1997 (25.9.1997), totality.	1-32
A	BS2000/OSD-BC V1.0, Dynamic Binder Loader/Starter, Chapter 2.3.1: Unloading and unlinking objects (p.44), ID: U5137-J-Z125-2-7600 [online] April 1993 [retrieved on 1999-12-13]. Retrieved from the Internet via <URL:http://manuals.mchp.siemens.de/servers/bs2_man/man_us/bs2_bc/ V1_0/bls.pdf>	1-10,31,32
A	openUTM V4.0 (BS2000/OSD), Concepts and Functions, Chapter 5: Structure of UTM Applications, especially 5.2: The process concept, para. 3, page 67 , ID: U20683-J-Z135-2-7600 [online] February 1997 [retrieved on 1999-12-13]. Retrieved from the Internet via <URL: http:// manuals. mchp. siemens.de/servers/bs2_man/man_us/utm/v4_0/ utm_kon.pdf>	1-10,31,32
A	BS2000/OSD-BC V1.0, Performance Handbook, Chapter 5.3.1: Managing the resource main memory, especially: Deactivation (p.248), Waiting Time runout Control (p.255),Paging management algorithms (pp.256-257), ID: U1794-J-Z125-6-7600 [online] February 1997 [retrieved on 1999-12-13]. Retrieved from the Internet via <URL: http://manuals.mchp.siemens.de/servers/bs2_man/man_us /bs2_bc/V1_0/perform.pdf>	1-10,31,32

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/SG 99/00018

Patent document cited in search report			Publication date	Patent family member(s)			Publication date
US	A	5603031	11-02-1997	AU	A1	72164/94	06-02-1995
				EP	A2	634719	18-01-1995
				EP	A3	634719	03-01-1996
				JP	A2	7182174	21-07-1995
				JP	T2	7509799	26-10-1995
				WO	A1	9502219	19-01-1995
				US	A	6016393	18-01-2000
WO	A	9735262a				none	

PATENT COOPERATION TREATY

PCT

INTERNATIONAL SEARCH REPORT

(PCT Article 18 and Rules 43 and 44)

Applicant's or agent's file reference	FOR FURTHER ACTION	see Notification of Transmittal of International Search Report (Form PCT/ISA/220) as well as, where applicable, item 5 below.
FP 1103		
International application No.	International filing date (*day/month/year*)	(Earliest) Priority Date (*day/month/year*)
PCT /SG98/00102	16 December 1998 (16.12.98)	
Applicant		
Kent Ridge Digital Labs et al.		

This international search report has been prepared by this International Searching Authority and is transmitted to the applicant according to Article 18. A copy is being transmitted to the International Bureau.

This international search report consists of a total of 7 sheets.

☐ It is also accompanied by a copy of each prior art document cited in this report.

1. **Basis of the report**
   a. With regard to the **language**, the international search was carried out on the basis of the international application in the language in which it was filed, unless otherwise indicated under this item.
      ☐ the international search was carried out on the basis of a translation of the international application furnished to this Authority (Rule 23.1(b)).
   b. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, the international search was carried out on the basis of the sequence listing:
      ☐ contained in the international application in written form.
      ☐ filed together with the international application in computer readable form.
      ☐ furnished subsequently to this Authority in written form.
      ☐ furnished subsequently to this Authority in computer readable form.
      ☐ the statement that the subsequently furnished written sequence listing does not go beyond the disclosure in the international application as filed has been furnished.
      ☐ the statement that the information recorded in computer readable form is identical to the written sequence listing has been furnished.
2. ☐ **Certain claims were found unsearchable** (See Box I).
3. ☒ **Unity of invention is lacking** (See Box II).
4. With regard to the **title**,
   ☒ the text is approved as submitted by the applicant.
   ☐ the text has been established by this Authority to read as follows:
5. With regard to the **abstract**,
   ☒ the text is approved as submitted by the applicant.
   ☐ the text has been established, according to Rule 38.2(b), by this Authority as it appears in Box III. The applicant may, within one month from the date of mailing of this international search report, submit comments to this Authority.
6. The figure of the **drawings** to be published with the abstract is Figure No.: 1
   ☒ as suggested by the applicant. ☐ None of the figures.
   ☐ because the applicant failed to suggest a figure.
   ☐ because this figure better characterizes the invention.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SG 98/00102

Box I Observations where certain claims were found unsearchable (Continuation of item 1 of first sheet)

This International search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This international Searching Authority found multiple inventions in this international application, as follows:

see extra sheet

1. ☒ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying and additional fee, this Authority did not invite payment of any additional fee.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest.

☐ No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT

International application No.
PCT / SG 98/00102

A. CLASSIFICATION OF SUBJECT MATTER

IPC[6]: G06F 9/46, G06F 9/54

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC[6]: G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPODOC, WPI, The INTERNET, ACM Digital Library

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5603031 A (HELGESON et al.), 11 February 1997 (11.02.77) summary; column 7, line 66 - column 8, line 10; column 8, line 51 ff.; column 10, line 31- column 11, line 28; fig. 24A-C;	1-5,22-25,27-31,48-51
Y	totality.	5,12,31,38
X	Aptiva Benutzerhandbuch. MK NLS Center, Kst. 2877, June 1996, which is a translation of the Aptiva Handbook, IBM Nr. 07H1639, International Business Machines Corporation "Hinweise zu den Rapid Resume Funktionen".	1-3,22,23,26-29,48,49,52
X	Rapid Resume. [online], [retrieved on 2000-02-16]. Retrieved from the Internet URL: <http://servicepac. mainz.ibm.com /eprmhtml /eprm3/7732.htm>; totality.	1-3,22,23,26-29,48,49,52

☒ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:
„A" document defining the general state of the art which is not considered to be of particular relevance
„E" earlier application or patent but published on or after the international filing date
„L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
„O" document referring to an oral disclosure, use, exhibition or other means
„P" document published prior to the international filing date but later than the priority date claimed
„T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
„X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
„Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
„&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 May 2000 (19.05.00)	22 May 2000 (22.05.00)
Name and mailing adress of the ISA/AT Austrian Patent Office Kohlmarkt 8-10; A-1014 Vienna Facsimile No. 1/53424/535	Authorized officer Fastenbauer Telephone No. 1/53424/447

Form PCT/ISA/210 (second sheet) (July 1998)

INTERNATIONAL SEARCH REPORT

International application No.
PCT /SG 98 / 00102

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	OSHIMA Mitsuru and KARJOTH Guenther. Aglets Specification (1.0), Draft 0.30 [online]1997-05-20 [retrieved on 2000-05-18]. Retrieved from the Internet URL:< http://www.trl.ibm.co.jp/aglets/spec10.html>	1-4,6-12,14,16, 19-30,32-38, 40,42,45,48-52
Y	totality.	5,12,31,38
A	CHAUHAN Deepika and BAKER Albert D. JAFMAS: a multiagent application development system. Proceedings of the second international conference of Autonomous agents, Minneapolis, MN USA, p. 100-107 [online], May 10-13,1998 [retrieved on 2000-05-17]. Retrieved from the Internet URL:< http://www.acm.org/pubs/citations/ proceedings /ai/280765/p100-chauhan/p100-chauhan.pdf>; totality.	1-52
A	VENNERS Bill. Solve real problems with aglets, a type of mobile agent. JavaWorld, May 1997 [online] [retrieved on 2000-05-18]. Retrieved from the Internet URL:< http: http://www.javaworld.com/javaworld/jw-05-1997/f_jw-05-hood.html>; totality, especially Aglets, A refresher	1-52
A	ARIDOR, Yariv, and LANGE Danny B. Agent design patterns: elements of agent application design. Proceedings of the second international conference of Autonomous agents, Minneapolis, MN USA, p.108-115 [online], May 10-13,1998 [retrieved on 2000-05-17]. Retrieved from the Internet URL:< http://www.acm.org/pubs/articles/proceedings/ai/280765/p108-aridor/p108-aridor.pdf>; totality.	1-52
A	openUTM V4.0 (BS2000/OSD), Concepts and Functions, ID: U20683-J-Z135-2-7600 [online] February 1997 [retrieved on 1999-12-13]. Retrieved from the Internet via <URL: http://manuals.mchp.siemens.de/servers /bs2_man /man_us/utm/v4_0/utm_kon.pdf> Chapter 5: Structure of UTM Applications, especially 5.2: The process concept, para. 3, page 67 .	1-52
A	BS2000/OSD-BC V1.0, Performance Handbook, ID: U1794-J-Z125-6-7600 [online] February 1997 [retrieved on 1999-12-13]. Retrieved from the Internet via <URL: http://manuals.mchp.siemens.de/servers/bs2_man/man_us/bs2_bc/V1_0/perform.pdf> Chapter 5.3.1: Managing the resource main memory, especially: Deactivation (p.248), Waiting Time runout Control (p.255),Paging management algorithms (pp.256-257).	1-52

Form PCT/ISA/210 (continuation of second sheet) (July 1998)

:- contained in 023/98/UBQ/PL.

INTERNATIONAL SEARCH REPORT

International application No.
PCT /SG 98 / 00102

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	
A	BS2000/OSD-BC V1.0, Dynamic Binder Loader/Starter, ID: U5137-J-Z125-2-7600 [online] April 1993 [retrieved on 1999-12-13]. Retrieved from the Internet via <URL: http://manuals.mchp.siemens.de/servers/bs2_man/man_us/bs2_bc/V1_0/bls.pdf> Chapter 2.3.1: Unloading and unlinking objects (p.44).	1-52
A	WO 97/35262 A (HITACHI), 25 September 1997 (25.09.97), summary.	1-52

Form PCT/ISA/210 (continuation of second sheet) (July 1998)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SG 98/00102

new				
1	cl. 1-4	+	cl. 27-28 cl. 30	a computing environment (method) wherein a process ... is treated as an entity that can be transferred ... a construct is formed ... / suspend all active threads ... falling whithin lists
2			cl. 29	... comprises all the data ...
3	cl. 5	+	cl. 31	... authorizing signature
4	cl. 6	+	cl. 32	... selective deletion of objects ...
5	cl. 7	+	cl. 33	... selective loading / Reloading of objects ...
6	cl. 8	+	cl. 34	... includes ... the incorporation ... of new objects ...
	cl. 9	+	cl. 35	... construct ... comprising at least ... data and/or program modules ... transferred to said first process ... stores said new process ...
	cl. 10	+	cl. 36	... suspends all active threads ... and creates an new process ...
	cl. 11	+	cl. 37	... falling within lists ...
	cl. 12	+	cl. 38	... authorizing signature
	cl. 13	+	cl. 39	... after said construct is transferred ... the second process is caused to be activeted ...
	cl. 14	+	cl. 40	... the first process is suspended and the second ... activated ... the first re-activated ...
	cl. 15	+	cl. 41	... some of the data / modules ... are added to the second process ..
	cl. 16	+	cl. 42	... a construct is formed ... and transferred to said first process
	cl. 17	+	cl. 43	... suspends all active threads of said second process and creates a new process ...
	cl. 18	+	cl. 44	... whithin lists...
	cl. 19	+	cl. 45	... said data and said program modules are copied into said first process
7	cl. 22	+	cl. 48	said process may be transferred ...
	cl. 23	+	cl. 49	... a construct is formed at least some of the data and/or program modules and execturion states
	cl. 24	+	cl. 50	... a subset of the data, program modules and execution states ...
	cl. 25	+	cl. 51	... allows to run in the second hardware ...
	cl. 26	+	cl. 52	... is a memory storage device ...

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/SG 98/00102

Patent document cited in search report			Publication date	Patent family member(s)			Publication date
US	A	5603031	11-02-1997	AU	A1	72164/94	06-02-1995
				EP	A2	634719	18-01-1995
				EP	A3	634719	03-01-1996
				JP	A2	7182174	21-07-1995
				JP	T2	7509799	26-10-1995
				WO	A1	9502219	19-01-1995
				US	A	6016393	18-01-2000
WO	A1	9735262	25-09-1997			none	

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 9/46, 9/54

A1

(11) International Publication Number: WO 00/36508

(43) International Publication Date: 22 June 2000 (22.06.00)

(21) International Application Number: PCT/SG99/00009

(22) International Filing Date: 22 February 1999 (22.02.99)

(30) Priority Data:
PCT/SG98/00102 16 December 1998 (16.12.98) SG

(71) Applicant *(for all designated States except US)*: KENT RIDGE DIGITAL LABS [SG/SG]; 21 Heng Mui Keng Terrace, Singapore 119613 (SG).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: NGAIR, Teow, Hin [SG/SG]; 334 Kang Ching Road #13–254, Singapore 610334 (SG). PANG, Hwee, Hwa [SG/SG]; 19 Shelford Road #01–42, Singapore 288408 (SG).

(74) Agent: GREENE–KELLY, James, Patrick; Lloyd Wise, Tanjong Pagar, P.O. Box 636, Singapore 910816 (SG).

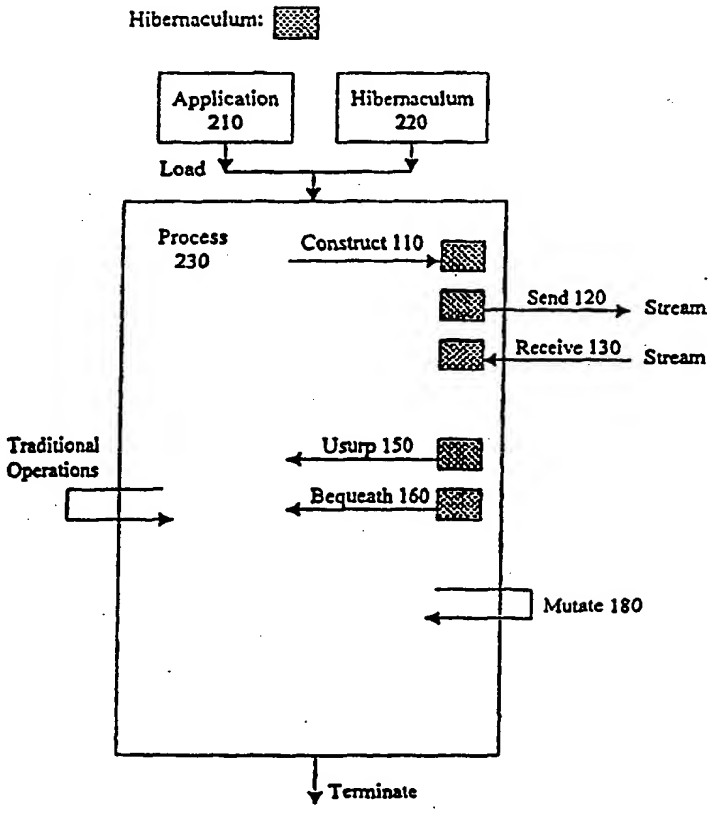
(81) Designated States: JP, SG, US.

Published
With international search report.

(54) Title: A METHOD FOR DETACHING AND RE–ATTACHING COMPONENTS OF A COMPUTING PROCESS

(57) Abstract

A method is described for detaching and then later re–attaching components of a computer process in which a process is split into a first process and a second sub–process. The sub–process may be a dormant process containing data, program modules and execution states not immediately required by the active first process. The dormant process is stored in a construct that may be kept in the computing device or may be sent to an external memory device. Alternatively the sub–process may comprise a permanently unwanted sub–process that is to be discarded. The invention allows maximum usage of limited resource computing systems.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

A METHOD FOR DETACHING AND RE-ATTACHING COMPONENTS OF A COMPUTING PROCESS

This invention relates to a method for detaching and re-attaching components of a computing process, and in particular for example to such a method for removing parts of the data, program code and execution state of a process and for transferring those parts to secondary or tertiary storage where they may be stored while not needed, before being re-attached to the process when required once more.

In a number of computing environments a large number of processes may be running concurrently on a single machine. This can happen for example in a follow-me computing environment where processes can span multiple user sessions. This results in a large number of processes – some active, some suspended – on a single machine. The presence of such a large number of concurrent processes can place a heavy burden on the resources of the system and can substantially slow down the running speed of the system.

A similar problem can occur in networks involving computing devices that may have limited memory space. Unless the memory space of such devices is used with maximum efficiency, the memory can quickly become overloaded. Minimizing resource usage is therefore imperative.

It would therefore be desirable if it were possible for elements of a computing process that were temporarily not required to be removed from the limited device or network and stored in such a way that they could be re-integrated into the process at a later time when required.

Recent years have seen a number of developments in computing science regarding how elements within a software application are treated and handled. In this context the most basic elements to be found within a software application are data and program modules. Traditional procedural programming paradigms focus on the logic of the software application, so a program is structured based on program modules. One uses the program modules by explicitly supplying to them the data on which they should operate.

More recently there has been a move towards an object-oriented paradigm. In this paradigm, programs are structured around objects, with each object representing an entity

in the world being modeled by the software application. Each object manages its own data (state), which are hidden from the external world (other objects and programs). An object in a program interacts with another object by sending it a message to invoke one of its exposed program modules (method). This paradigm imposes better control and protection over internal data, and helps to structure complex applications designed and implemented by a team of programmers. An example of an object-oriented environment can be found in US 5,603,031. This discloses an environment in which new agents (essentially objects) consisting of data and program modules can be sent between machines. However, in the environment of US 5,603,031 a special application code is needed to convert state information into data in the newly created process, and to ensure that the latter begins execution from the right instruction. Since execution state cannot be manipulated directly, it is very tedious, if not impossible, to transfer shared resources within a process to a surrogate. Moreover since the surrogate has a different identity from the original process, shared resources managed externally, such as data locks, cannot be transferred.

While object-oriented paradigm represents a significant advance in software engineering, the data and modules that constitute each object are static. The paradigm is still inadequate for writing programs that must evolve during execution, eg programs that need to pick up, drop, or substitute selected modules. There have been several attempts at overcoming this limitation. For example, work described in US Patents 4954941, 5175828, 5339430 and 5659751 address techniques for re-linking or re-binding selected software modules dynamically during runtime. Also Microsoft's Win32 provides for explicit mapping and un-mapping of dynamic linked libraries into the address space of a process through the LoadLibrary and FreeLibrary calls. With this prior art, however, the prototype or specification of functions and symbols are fixed beforehand and compiled into application programs. This enables a process to mutate into and back from a surrogate by replacing selected program modules. However, the replacement modules must retain the same function prototypes making coding complicated and unnatural. Also data structures that are not needed by the surrogate remain in memory.

Work has also been done in the area of process checkpointing and recovery. This deals with mechanisms for preserving process state and recovery from machine crashes.

The mechanisms operate from outside of the process and the process remains the same upon resumption.

In this specification the following terms will be used with the following meaning:

"First class entity": an object that can be manipulated directly.

"Process": a combination of data, program module(s) and current execution state.

"Execution state": the values and contents of transient parameters such as the contents of registers, frames, counters, look-up tables and the like.

According to the present invention there is provided a method for removing a part of a computing process, wherein said process splits into a first process and a sub-process, the sub-process comprising items of data and/or program code and/or execution states of said computing process temporarily not required by said first process.

By means of this arrangement a process is enabled to temporarily or permanently detach from itself sub-processes comprising subsets of the data, program code and execution states of the process, and to remove the sub-process to, for example, secondary or tertiary storage until they are needed again. This compacts the size of the process, freeing up resources for other processes. If the sub-process is not required again it may be discarded completely. The sub-process may comprise only data, or only program code or only execution state information, or any combination of these three components.

The method of the present invention may be implemented by two approaches. In a first approach the process splits into a first process and sub-processes with the first process retaining the process identity of the original computing process. Alternatively, if retaining original process identity is not required, the original process may split into two new processes.

In either event the sub-process may comprise data, program code and execution states that is permanently not required and which may be deleted, or it may comprise data, program code and execution states that is only temporarily not required and which may be partially or totally re-acquired by the first process. Data, program code and execution states may be temporarily not required by the first process generally, or may be specific to a particular user of computing means who may be temporarily absent from the computing means. In either case temporarily removing the sub-process frees up resources for the first process.

The method is founded upon the ability of a process to manipulate its own data, program code and execution state directly. This ability enables a process to split itself into a first process and a second sub-process.

In a preferred embodiment a construct is formed for storing the sub-process. This construct may be formed by a construct operation that suspends all active threads of the computing process and creates a new process comprising at least some of the data and/or program modules and/or execution state of the computing process, and stores the new process in a data area of the computing process.

In a preferred embodiment the construct comprises only data, program modules and execution state falling within lists that are passed to the construct operation.

Optionally the construct may be provided with an authorising signature.

If desired, following formation of a construct storing the sub-process the construct may be sent to a memory storage device from which the construct may be retrieved when required once more.

While running the first process may re-acquire data, program codes and execution states from the sub-process as and when required. The first process may also load selected program modules as required.

When it is time to restore the original process, the first process may simply drop off any unwanted extraneous data, program code and execution states that it may have picked up during execution, and then merges with the sub-process to pool together their data, program code and execution states.

An embodiment of the invention will now be described by way of example and with reference to the accompanying drawings, in which:-

Fig.1 is a general schematic model of an operating environment of a computing system,

Fig.2 schematically illustrates a process life-cycle and operations that may be carried out on the process,

Fig.3 is a flowchart illustrating the hibernaculum Construct operation,

Fig.4 is a flowchart illustrating the Mutate operation,

Fig.5 is a flowchart illustrating the Usurp operation, and

Fig.6 is a flowchart illustrating the Bequeath operation.

Figure 1 shows the general model of a computing system. An application program 30 comprises data 10 and program modules 20. The operating system 60, also known as the virtual machine, executes the application 30 by carrying out the instructions in the program modules 20, which might cause the data 10 to be changed. The execution is effected by controlling the hardware of the underlying machine 70. The status of the execution, together with the data and results that the operating system 60 maintains for the application 30, form its execution state 40.

Such a model is general to any computing system. It should be noted here that the present invention starts from the realisation that all the information pertaining to the application at any time is completely captured by the data 10, program modules 20 and execution state 40, known collectively as the process 50 of the application 30.

The process 50 can have one or more threads of execution at the same time. Each thread executes the code of a single program module at any given time. Associated with the thread is a current context frame, which includes the following components:

- A set of registers
- A program counter, which contains the address of the next instruction to be executed
- Local variables of the module
- Input and output parameters of the module
- Temporary results of the module

In any module A, the thread could encounter an instruction to invoke another module B. In response, the program counter in the current frame is incremented, then a new context frame is created for the thread before it switches to executing module B. Upon completing module B, the new context frame is discarded. Following that, the thread reverts to the previous frame, and resumes execution of the original module A at the instruction indicated by the program counter, i.e., the instruction immediately after the module invocation. Since module B could invoke another module, which in turn could invoke some other module and so on, the number of frames belonging to a thread may grow and reduce with module invocations and completions. However, the current frame of a thread at any given time is always the one that was created last. For this

reason, the context frames of a thread are typically stored in a stack with new frames being pushed on and popped from the top. The context frames of a thread form its execution state, and the state of all the threads within the process 50 constitute its execution state 40 in Fig.1.

The data 10 and program modules 20 are shared among all threads. The data area is preferably implemented as a heap, though this is not essential. The locations of the data 10 and program modules 20 are summarized in a symbol table. Each entry in the table gives the name of a datum or a program module, its starting location in the address space, its size, and possibly other descriptors. Instead of having a single symbol table, each process may alternatively maintain two symbol tables, one for data alone and the other for program modules only, or the process could maintain no symbol table at all.

In a preferred embodiment of the present invention, the data and program code of a process are stored in a heap and a program area respectively and are shared by all the threads within the process. In addition the execution state of the process comprises a stack for each thread, each stack holding context frames, in turn each frame containing the registers, local variables and temporary results of a program module, as well as addresses for further module invocations and returns. Before describing an embodiment of the invention in more detail, however, it is first necessary to introduce some definitions of data types and functions that are used in the embodiment and which will be referred to further below.

In addition to conventional data types such as integers and pointer, four new data types Data, Module, Stack and Hibernaculum are defined in the present invention:

Data: A variable of this data type holds a set of data references. Members are added to and removed from the set by means of the following functions;

Int AddDatum(Data d, String dataname) inserts the data item dataname in the heap of the process as a member of d.

Int DelDatum(Data d, String dataname) removes the data item dataname from d.

Module: A variable of this data type holds a set of references to program modules. Members are added to and removed from the set with the following functions;

Int AddModule(Module d, String modulename) inserts the program module modulename in the program area of the process as a member of d.

Int DelModule(Module d, Stringname modulename) removes the program module modulename from d.

Stack: A variable of this data type holds a list of ranges of execution frames from the stack of the threads. The list may contain frame ranges from multiple threads, however no thread can have more than one range. Variables of this type are manipulated by the following functions:

Int OpenFrame(Stack d, Thread threadname) inserts into d a new range for the thread threadname, beginning with the thread's current execution frame. This function has no effect if the thread already has a range in d.

Int CloseFrame(Stack d, Thread threadname) ends the open-ended range in d that belongs to the thread threadname. This function has no effect if the thread does not currently have an open-ended range in d.

Int PopRange(Stack d, Thread threadname) removes from d the range belonging to the thread threadname.

Hibernaculum: A variable of this data type is used to hold a suspended process.

As will be explained in more detail below a process may be suspended and stored in a construct formed by the following operation:

Hibernaculum Construct(Stack s, Module m, Data d): This operation creates a new process with the execution state, program table and data heap specified as input parameters. The process is immediately suspended and then returned in a hibernaculum.

The hibernaculum may be signed by the originating process as indication of its authenticity. Fig.3 is a flow-chart showing the hibernaculum construct operation.

A hibernaculum may be sent between operating environments by the following send and receive functions:

Int Send(Hibernaculum h, Target t) transmits the process contained within h to the specified target.

Hibernaculum Receive(Source s) receives from the specified source a hibernaculum containing a process.

A hibernaculum may be subject to the following functions:

Int Usurp(Hibernaculum h, OverrideFlags f) copies the data and program modules of the process within h into the calling process's operating environment. Where there is a conflict between the data and/or program modules of the hibernaculum and the operating environment, the override flags specify which to preserve. Fig.5 is a flow-chart illustrating the steps of the usurp operation.

Int Bequeath(Hibernaculum h, Thread threadname, OverrideFlags f), upon threadname's termination, activates the threads of the process stored within h and runs them as threads within the environment of the process containing threadname. Where there is a conflict between the data and/or program modules of the hibernaculum and the calling process, the override flags specify which is to be preserved. Fig.6 is a flow-chart illustrating the steps of the bequeath operation.

Int Mutate(Stack s, int sFlag, Module m, int mflag, Data d, int dflag) modifies the execution state, program table and data heap of the calling process. If a thread has an entry in s, only the range of execution frames specified by this entry is preserved, the

other frames are discarded. Execution stacks belonging to threads without an entry in s are left untouched. In addition, program modules listed in m and data items listed in d are kept or discarded depending on the flag status. Fig.4 is a flow-chart illustrating the steps of the mutate operation.

In the following exemplary embodiment of the invention we shall assume that a user of a computing device is about to leave his machine and wishes to remove parts of a process to secondary or tertiary storage, while retaining enough of the data, program code and execution states to manage and respond to events on resources (eg network sockets and data locks) that are shared with the external world (eg other processes). Removing temporarily unwanted sub-processes frees up resources for other processes and/or users.

In the example that follows a number of operations that act upon a process are invoked by the process. These operations are Construct, Mutate, Usurp and Bequeath. Before describing the example in detail these operations – which are schematically illustrated in Fig.2 - will be discussed.

New processes are created with a Construct 110 operation. Fig.3 is a flow-chart showing the steps of this Construct operation. Each invocation of this operation starts up a controller thread in the process 230. The controller thread freezes all other active threads in the process 230, then creates a new process with some or all of the execution state, program modules and/or data of the process 230 except for those belonging to the controller thread, before resuming the frozen threads. Therefore, the new process contains no trace of the controller thread. The new process is suspended immediately and returned in a hibernaculum in the data area of the process 230. As explained earlier, a hibernaculum is a special data type that serves the sole purpose of providing a container for a suspended process. Since a process may have several hibernacula in its data area, it could create a new hibernaculum that contains those hibernacula, each of which in turn could contain more hibernacula, and so on. When the new process is activated subsequently, only those threads that were active just before the Construct 110 operation will begin to execute initially; threads that were suspended at that time will remain suspended. At the end of the Construct 110 operation, the controller thread resumes those threads that were frozen by it before terminating itself.

To specify what execution state should go into the new process, the Construct 110 operation is passed a list of ranges of context frames. The list may include frames from the state of several threads. No thread is allowed to have more than one range in this list. A thread can specify that all of its frames at the time of the Construct 110 operation are to be included in the list, by calling the AllFrame function beforehand. Alternatively, the thread can call the OpenFrame function to register its current frame, so that all the frames from the registered frame to the current frame at the time of the Construct 110 operation are included in the list. The thread can also call the CloseFrame function subsequently, to indicate that only frames from that registered with OpenFrame to the current frame at the time of calling CloseFrame are to the included in the list for the Construct 110 operation. An AllFrame or OpenFrame request for a thread erases any previous AllFrame, OpenFrame and CloseFrame requests for that thread. A CloseFrame request overrides any earlier CloseFrame request for the same thread, but the request is invalid if the thread has not already had an OpenFrame request. A thread can also make AllFrame, OpenFrame and/or CloseFrame requests on behalf of another thread by providing the identity of that thread in the requests; the effect is as if that thread is making those requests itself.

The Construct 110 operation can also be passed a list of program modules that should go into the newly created process. A thread can specify that all modules of the process 230 are to be included in the list, by calling the AllModules function prior to the Construct 110 operation. Alternatively, the thread can call the AddModule function to add a specific module to the list, and the DelModule function to remove a specific module from the list. The effect of the AllModules, AddModule and DelModule requests, possibly made by different threads, are cumulative. Hence a DelModule request after an AllModules request would leave every module in the list except for the one removed explicitly, and a DelModule can be negated with an AddModule or AllModules request. As there could be multiple AddModule requests for the same module and AllModules could be called multiple times, a program module may be referenced several times in the list. However, the Construct 110 operation consolidates the entries in the list, so no program module gets duplicated in the new process.

To copy some or all of the data of the process 230 to the new process, the Construct 110 operation can be passed a data list. This list contains only the name of, or reference to data that should be copied. The actual data content or values that get copied to the new process are taken at the time of the Construct 110 operation, not at the time that each datum is added to the list. To ensure consistency among data that could be related to each other, all the threads in the process 230 are frozen during the Construct 110 operation. A thread can specify that all data of the process 230 are to be included in the list, by calling the AllData function prior to the Construct 110 operation. Alternatively, the thread can call the AddDatum function to add a specific datum to the list, and the DelDatum function to remove a specific datum from the list. The effect of the AllData, AddDatum and DelDatum requests, possibly made by different threads, are cumulative. Hence a DelDatum request after an AllData request would leave all of the data in the list except for the one removed explicitly, and a DelDatum can be negated with an AddDatum or AllData request. As there could be multiple AddDatum requests for the same datum and AllData could be called multiple times, a datum may be referenced several times in the list. However, the Construct 110 operation consolidates the entries in the list, so no datum gets duplicated in the new process.

Since the lists passed to the Construct 110 operation are constructed from the execution state, program modules and data of the process 230, the new process initially does not contain any component that is not found in the process 230. Consequently, the symbol table in the new process is a subset of the symbol table of the process 230. Threads in the process 230 that do not have any frame in the new process are effectively dropped from it. For those threads that have frames in the new process, when activated later, each will begin execution at the instruction indicated by the program counter in the most recent frame amongst its frames that are copied. By excluding one or more of the most recent frames from the new process, the associated thread can be forced to return from the most recent module invocations. An exception is raised to alert the thread that those modules are not completed normally. Alternatively, the thread can be made to redo those modules upon activation, by decrementing the program counter in the most recent frame amongst those frames belonging to that thread that are copied. Similarly, by

excluding one or more of its oldest frames from the new process, a thread can be forced to terminate directly after completing the frames that are included.

The process 230 can also modify any of its components directly by calling the Mutate 180 operation from any thread. Fig.4 shows the flow-chart for the Mutate operation. The operation starts up a controller thread in the process 230. The controller thread first freezes all other active threads in the process 230, then selectively retains or discards its execution state, program modules and data. A list of context frames, together with a flag, can be passed to the Mutate 180 operation to retain or discard the frames in the list. Similarly, a program module list and/or a data list can be provided to indicate program modules and data of the process 230 that should be retained or discarded. The generation of the execution state, program module and data lists are the same as for the Construct 110 operation. After mutation, the controller thread resumes the threads that were frozen by it before terminating itself. Threads that no longer have a context frame in the process 230 are terminated. Each of the remaining threads resumes execution at the instruction indicated by the program counter in the most recent frame amongst the retained frames belonging to that thread. By discarding one or more of its most recent frames, a thread can be forced to return from the most recent module invocations. An exception is raised to alert the thread that those modules are not completed normally. Similarly, by discarding one or more of its oldest frames, a thread can be forced to terminate directly after completing the frames that are retained. Space freed up from the discarded context frames, program modules and data is automatically reclaimed by a garbage collector.

The Usurp 150 operation, which also accepts a hibernaculum as input, enables the process 230 to take in only program modules and data from a hibernaculum, without acquiring its threads. Fig.5 is a flow-chart showing this operation in detail. The operation starts up a controller thread in the process 230. The controller thread freezes all other active threads in the process 230, adds the program modules and data of the process in the hibernaculum to the program modules and data of the process 230, respectively, and updates its symbol table accordingly. In case of a name conflict, the program module or data from the hibernaculum is discarded in favor of the one from the process 230 by default. However, a flag can be supplied to give preference to the hibernaculum. Finally,

all the original threads in the process 230 that were frozen by the controller thread are resumed before it terminates itself.

The Bequeath 160 operation accepts a hibernaculum and a thread reference as input. Fig.6 is a flow-chart showing this operation in detail. It starts up a bequeath-thread in the process 230. The bequeath-thread registers the referenced thread and any existing bequeath-threads on that thread, then allows them to carry on execution without change. After all those threads and threads that they in turn activate have terminated, the bequeath-thread loads in the context frames, program modules and data in the hibernaculum. In case of a name conflict, the program module or data from the hibernaculum is discarded in favor of the one from the process 230 by default. However, a flag can be supplied to give preference to the hibernaculum. Subsequently, threads within the hibernaculum are activated to run in the process 230 before the bequeath-thread terminates itself. Only those threads that were active just before the hibernaculum was created are activated initially; threads that were suspended at that time remain suspended. Each newly acquired thread begins execution at the instruction indicated by the program counter in the most recent frame amongst the context frames that belong to that thread. If multiple Bequeath 160 requests are issued for the same thread, there will be several bequeath-threads in the process 230. Each bequeath-thread will wait for all the existing bequeath-threads on the same thread, together with all the threads that they activate, to terminate before performing its function. As a result, the Bequeath 160 requests are queued up and serviced in chronological order.

A process stored in a hibernaculum may also be transmitted to another device (for example a storage device) by the Send operation 120 and similarly may be received by the receive operation 130.

To begin the process p1 that is to be split calls the function Hibernaculum Construct:

Hibernaculum h = Construct(Stack s, Module m, Data d)

where s, m, and d contain the execution stacks, program modules and data, respectively, in process p1 that are not required in order to respond to events on shared

resources. This function creates a new process p2 to hold those heap, program table and execution states. The new process p2 is immediately suspended and returned within a hibernaculum h.

This Hibernaculum Construct operation is shown in the flowchart of Fig.3 and the result is a construct that holds the temporarily non-required sub-process.

The process p1 then calls the mutate function:

Int Mutate(Stack –s, Module –m, Data –d)

which discards from p1 the execution stacks, program modules and data specified in s, m and d respectively (ie the data held in the hibernaculum h). As a result p1 now holds only those data, program modules and execution states of the original process that are needed to respond to events on shared resources. It should be noted that the mutated process retains the same process identity, ie p1, as the original process. This mutate operation is shown in the flowchart of Fig.4. It should be noted here that while in this embodiment the first or active process retains the process identity of the original process, this is not essential and in an alternative the Construct operation can be used to generate a first process having a new process identity.

The mutated process p1 loads in appropriate event handling modules, then waits for incoming events. As p1 executes, the process may load in additional program modules, or may acquire data, program modules or execution states from the dormant process p2 using the function:

Int Usurp(Hibernaculum h, Stack s, Module m, Data d)

to take in from p2, which resides in the hibernaculum h, the execution stack, the program modules and data specified by s, m and d (and which may be a subset of those parameters held in the hibernaculum h). The usurp operation is shown in the flowchart of Fig.5.

When the user returns to the machine and wishes to restore the original process the mutated process p1 calls the function:

Int Mutate(Stack s, Module m, Data d)

where s, m and d contain the execution stacks, program modules and data that must be passed back to the original process. This discards any extraneous data, program code and/or execution states loaded in the user's absence.

Next p1 calls the function:

Int Bequeath(Hibernaculum h)

and quits. This results in all the execution stacks, program modules and data remaining in p1 being added to the dormant process p2 in the hibernaculum, and in p2 then being activated when p1 terminates. The Bequeath operation is shown in the flowchart of Fig.6.

It should also be understood that the dormant process p2 while held in the hibernaculum h could, if desired, be transmitted to another device such as a memory storage device, by using the Send operation 120. The dormant process p2 will then be re-transmitted and received using the receive operation 130 prior to the dormant process p2 being reattached to the original process. This would allow the memory capacity of the original device to be fully utilised by not requiring it to store dormant processes which may instead be stored offline.

It will also be understood that while in the above embodiment the first process is described as an active process and the second as a dormant process at least part of which may be reacquired by the active process later, the second process may alternatively comprise data, program code and execution states that are to be permanently deleted from the computing process as they are no longer required.

To implement the method of detaching and reattaching processes of the present invention in a Java environment, a package called snapshot is introduced. This package contains the following classes, each of which defines a data structure that is used in the operations involved in the method:

```
public class Hibernaculum {
        ...
}

public class State {
        ...
}

public class Module {
        ...
}

public class Data {
        ...
}

public class Machine {
        ...
}
```

In addition, the package contains a Snapshot class that defines the migration and adaptation operations:

```
public class Snapshot {
        private static native void registerNatives();
        static {
                registerNatives();
        }

        public static native Hibernaculum Construct(State s, Module m, Data d);
        public static native int Send(Hibernaculum h, OutputStream o);
```

```
        public static native Hibernaculum Receive(InputStream i);
        public static native int Usurp(Hibernaculum h, int f);
        public static native int Bequeath(Hibernaculum h, int f);
        public static native int Mutate(State s, Module m, int mflag, Data d, int dflag);

        // This class is not to be instantiated
        private Snapshot() {
        }
}
```

The methods in the Snapshot class can be invoked from application code. For example:

```
try {
        if (snapshot.Snapshot.Construct(s, m, d) != null) {
                // hibernaculum has been created
        } else {
                // failed to create hibernaculum
        }
        catch(snapShot.SnapshotException e) {
                // Failed to create hibernaculum
        }
```

The operations are implemented as native codes that are added to the Java virtual machine itself, using the Java Native Interface (JNI). To do that, a Java-to-native table is first defined:

```
#define KSH "Ljava/snapshot/Hibernaculum;"
#define KSS "Ljava/snapshot/State;"
#define KSM "Ljava/snapshot/Module;"
```

```
#define KSD "Ljava/snapshot/Data;"

static JNINativeMethod snapshot_Snapshot_native_methods[] = {
        {
                "Construct",
                "("KSSKSMKSD")"KSH,
                (void*)Impl_Snapshot_Construct
        },
        {
                "Send",
                "("KSH"Ljava/io/OutputStream;)I",
                (void*)Impl_Snapshot_Send
        },
        {
                "Receive",
                "(Ljava/io/InputStream;)"KSH,
                (void*)Impl_Snapshot_Receive
        },

        {
                "Usurp",
                "("KSH"I)I",
                (void*)Impl_Snapshot_Usurp
        },
        {
                "Bequeath",
                "("KSH"I)I",
                (void*)Impl_Snapshot_Bequeath
        },

        {
```

```
            "Mutate",
            "("KSSKSM"I"KSD"I)I",
            (void*)Impl_Snapshot_Mutate
        },

};
```

After that, the native implementations are registered via the following function:

```
JNIEXPORT void JNICALL
Java_snapshot_Snapshot_registerNatives(JNIEnv *env, jclass cls) {
    (*env)->RegisterNatives(    env,
                                cls,
                                snapshot_Snapshot_native_methods,
                                sizeof(snapshot_Snapshot_native_methods) /
                                sizeof(JNINativeMethod)     );
}
```

Besides the above native codes, several functions are added to the Java virtual machine implementation, each of which realizes one of the migration and adaptation operations:

```
void* Impl_Snapshot_Construct(..) {
    // follow flowchart in Figure 3
    ...
}

void* Impl_Snapshot_Send(..) {
    // send given hibernaculum to specified target
    ...
}
```

```
void* Impl_Snapshot_Receive(..) {
        // receive a hibernaculum from a specified source
        ...
}

void* Impl_Snapshot_Usurp(..) {
        // follow flowchart in Figure 5
        ...
}

void* Impl_Snapshot_Bequeath(..) {
        // follow flowchart in Figure 6
        ...
}

void* Impl_Snapshot_Mutate(..) {
        // follow flowchart in Figure 4
        ...
}
```

CLAIMS

1. A method for removing a part of a computing process, wherein said process splits into a first process and a second sub-process, the second sub-process comprising items of data and/or program code and/or execution states of said computing process not required by said first process.

2. A method as claimed in claim 1 wherein said sub-process comprises items of data, program code and execution state of said computing process.

3. A method as claimed in claim 1 or 2 wherein a construct is formed for storing said sub-process, while said first process retains the process identity of said computing process.

4. A method as claimed in claim 3 wherein said construct is formed by a construct operation that suspends all active threads of said computing process and creates a new sub-process comprising at least some of the data and/or program modules and/or execution state of said computing process, and stores said sub-process in a data area of said computing process.

5. A method as claimed in claim 4 wherein said construct comprises only data, program modules and execution state falling within lists that are passed to said construct operation.

6. A method as claimed in any of claims 3 to 5 wherein said construct is provided with an authorising signature.

7. A method as claimed in any of claims 3 to 6 wherein said construct is sent to a memory storage device.

8. A method as claimed in any of claims 3 to 7 wherein said sub-process is a dormant process comprising data, program code and execution states of said computing process temporarily not required by said first process, and wherein said first process is able to re-acquire data, program codes and execution states from said dormant process as and when required by said first process.

9. A method as claimed in claim 8 wherein said first process re-acquires data, program code and execution states from said dormant process falling within specified ranges.

10. A method as claimed in claim 8 or 9 wherein said sub-process comprises data, program code and execution states specific to a given user of a computing means, and wherein said second process is formed to temporarily store said data, program code and execution states while said user is absent from said computing means.

11. A method as claimed in any of claims 3 to 10 wherein after said first process finishes executing data, program modules and execution states from said first process are added to said sub-process and said sub-process is reactivated.

12. A method as claimed in claim 11 wherein said first process discards extraneous data, program codes and execution states acquired subsequent to the formation of said first process and said sub-process prior to adding said first process to said sub-process.

13. A method as claimed in claim 1 or 2 wherein a construct is formed for storing said sub-process, while said first process is run in place of said computing process.

14. A method as claimed in claim 13 wherein said construct is formed by a construct operation that suspends all active threads of said computing process and creates a sub-process comprising at least some of the data and/or program modules and/or execution state of said computing process, and stores said sub-process in a data area of said first process.

15. A method as claimed in claim 14 wherein said construct comprises only data, program modules and execution state falling within lists that are passed to said construct operation.

16. A method as claimed in any of claims 13 to 15 wherein said construct is provided with an authorising signature.

17. A method as claimed in any of claims 13 to 16 wherein said construct is sent to a memory storage device.

18. A method as claimed in any of claims 13 to 17 wherein said sub-process is a dormant process comprising data, program code and execution states of said computing process temporarily not required by said first process, and wherein said first process is able to re-acquire data, program codes and execution states from said dormant process as and when required by said first process.

19. A method as claimed in claim 18 wherein said first process re-acquires data, program code and execution states from said dormant process falling within specified ranges.

20. A method as claimed in claim 18 or 19 wherein said sub-process comprises data, program code and execution states specific to a given user of a computing means, and wherein said sub-process is formed to temporarily store said data, program code and execution states while said user is absent from said computing means.

21. A method as claimed in any of claims 13 to 20 wherein after said first process finishes executing data, program modules and execution states from said first process are added to said sub-process and said sub-process is reactivated.

22. A method as claimed in claim 21 wherein said first process discards extraneous data, program codes and execution states acquired subsequent to the formation of said first process and said sub-process prior to adding said first process to said sub-process.

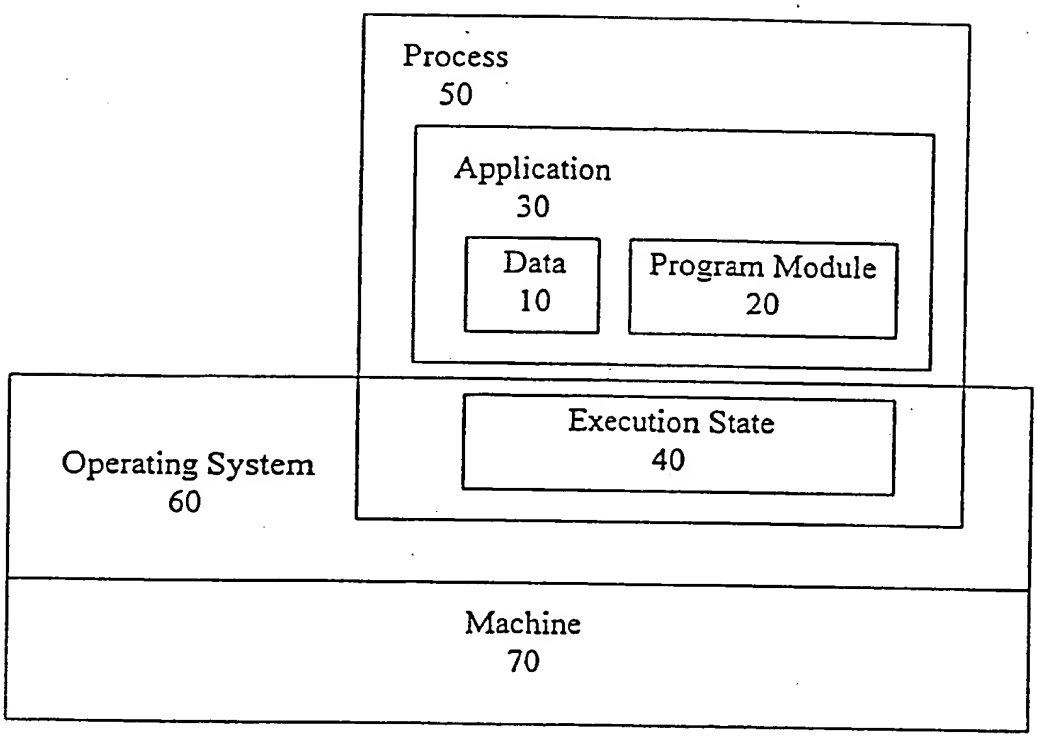

Process
50
Application
30
Data
10
Program Module
20
Execution State
40
Operating System
60
Machine
70

Fig.1

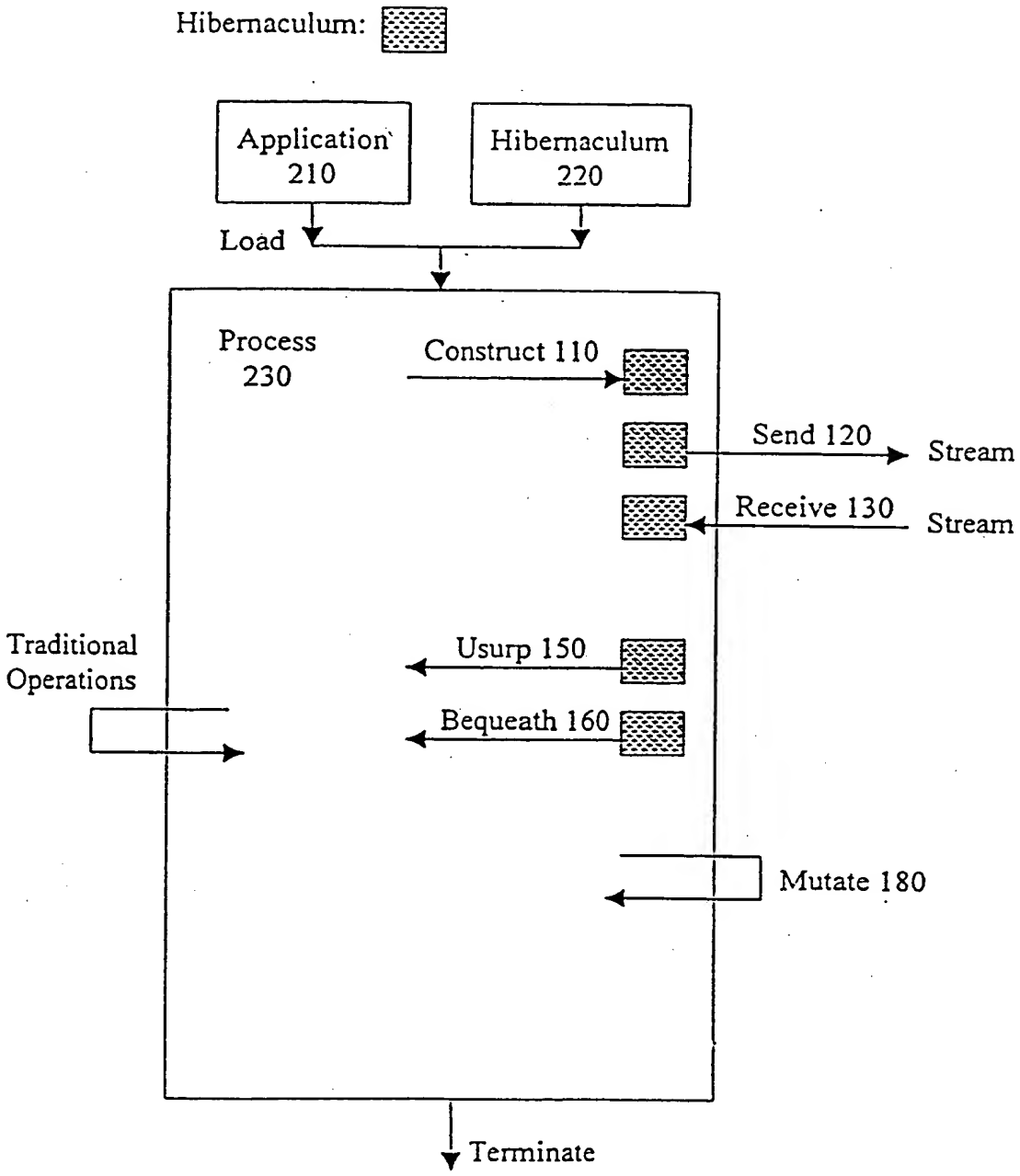

Hibernaculum:
Application 210
Hibernaculum 220
Load
Process 230
Construct 110
Send 120
Stream
Receive 130
Stream
Traditional Operations
Usurp 150
Bequeath 160
Mutate 180
Terminate

Fig.2

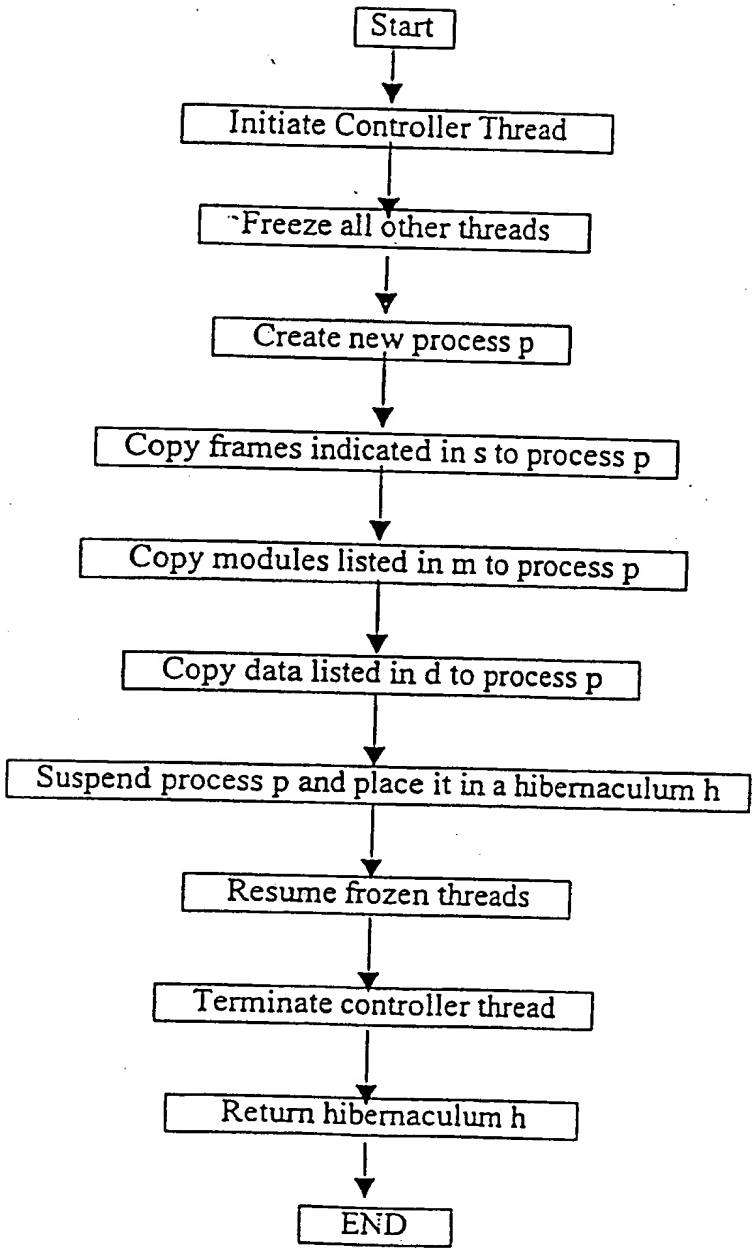

Start
Initiate Controller Thread
Freeze all other threads
Create new process p
Copy frames indicated in s to process p
Copy modules listed in m to process p
Copy data listed in d to process p
Suspend process p and place it in a hibernaculum h
Resume frozen threads
Terminate controller thread
Return hibernaculum h
END

Fig.3

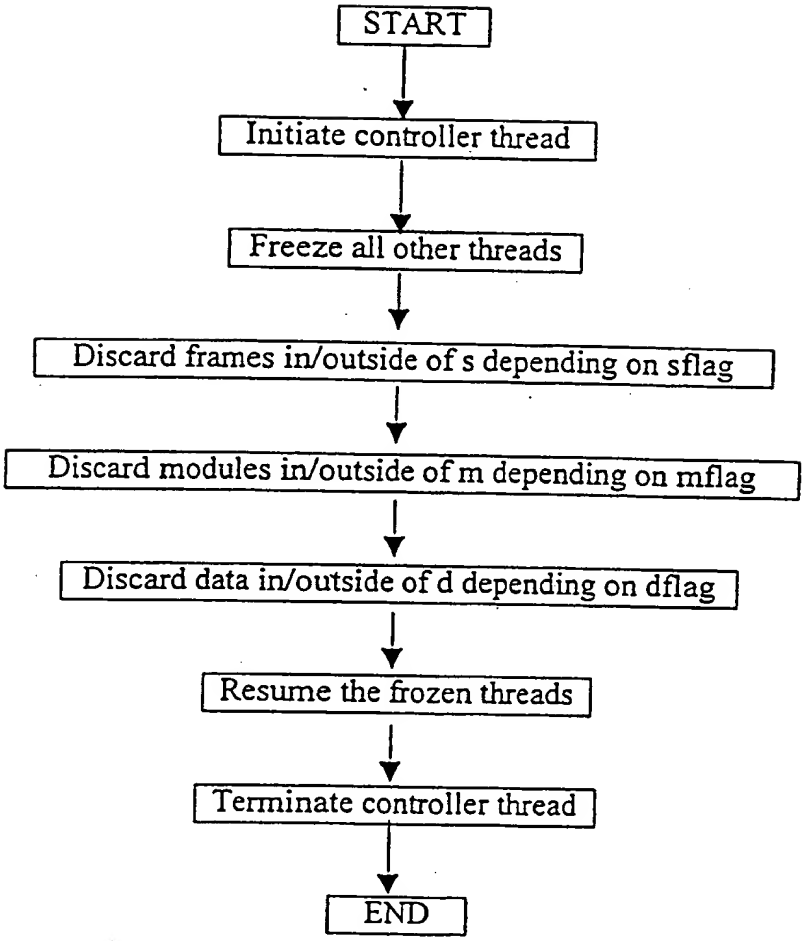

START
Initiate controller thread
Freeze all other threads
Discard frames in/outside of s depending on sflag
Discard modules in/outside of m depending on mflag
Discard data in/outside of d depending on dflag
Resume the frozen threads
Terminate controller thread
END

Fig.4

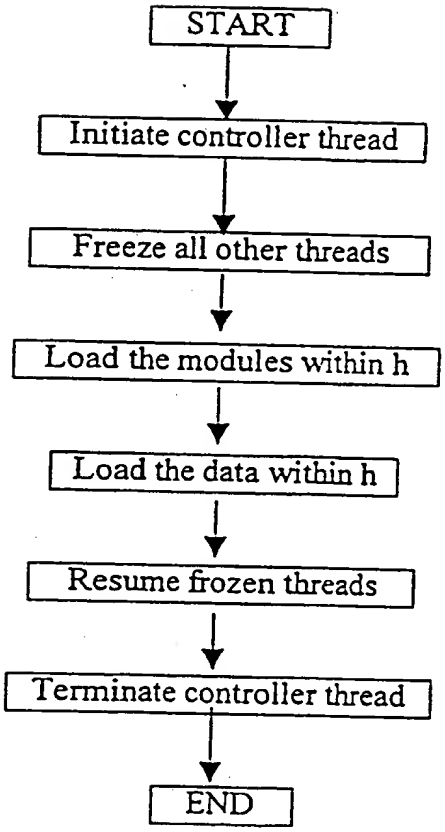

START
Initiate controller thread
Freeze all other threads
Load the modules within h
Load the data within h
Resume frozen threads
Terminate controller thread
END

Fig.5

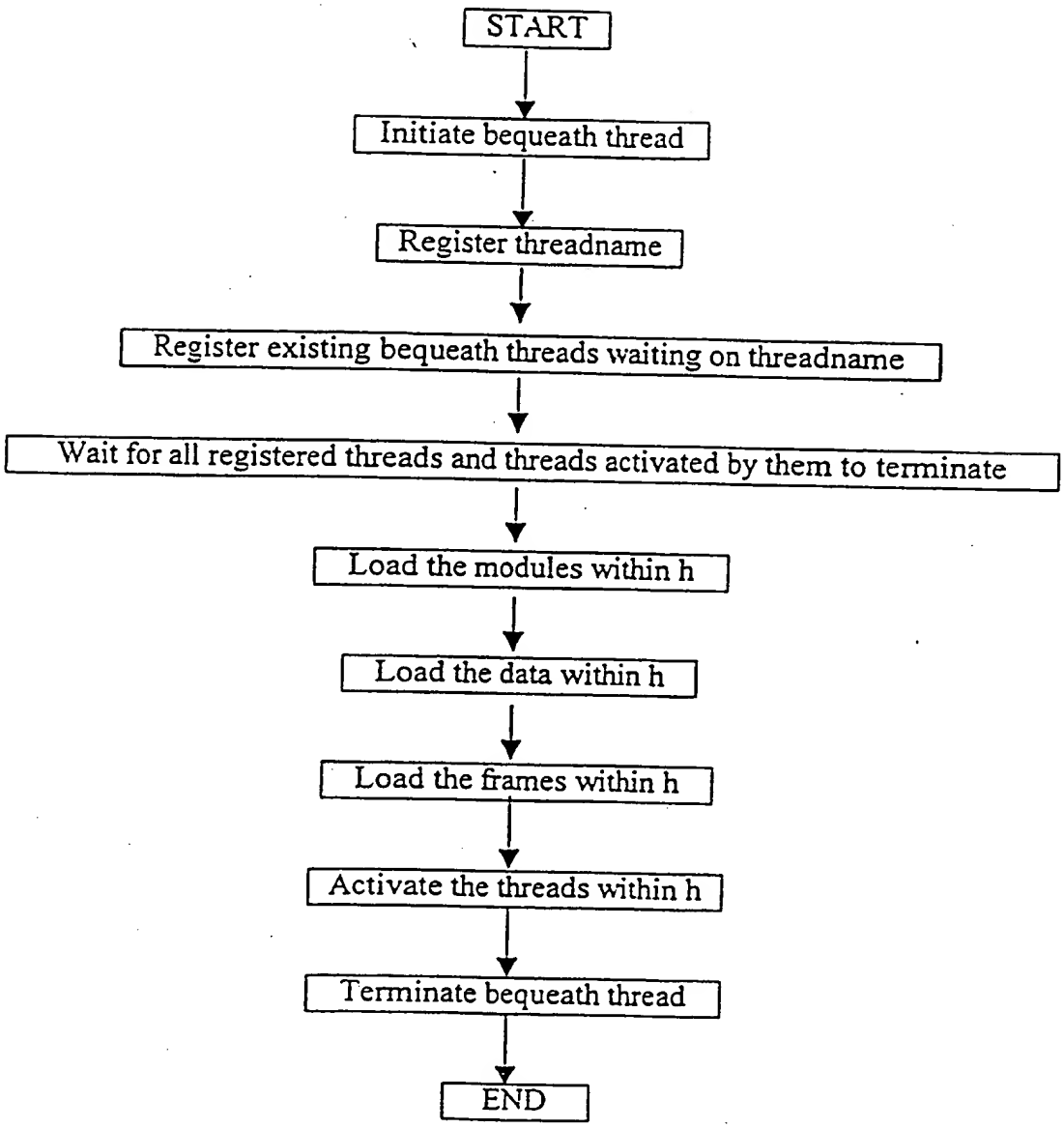

START
Initiate bequeath thread
Register threadname
Register existing bequeath threads waiting on threadname
Wait for all registered threads and threads activated by them to terminate
Load the modules within h
Load the data within h
Load the frames within h
Activate the threads within h
Terminate bequeath thread
END

Fig.6

INTERNATIONAL SEARCH REPORT

International application No.
PCT / SG 99/00009

A. CLASSIFICATION OF SUBJECT MATTER

IPC[6]: G06F 9/46, G06F 9/54

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC[6]: G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPODOC, WPI, The INTERNET

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X A	US 5603031 A (HELGESON et al.)11February 1997 (11.02.97) Summary of the Invention, para. 4,8,23; totality.	1-3,6 4-5,7-22
A	openUTM V4.0 (BS2000/OSD), Concepts and Functions, Chapter 5: Structure of UTM Applications, especially 5.2: The process concept, para. 3, page 67 , ID: U20683-J-Z135-2-7600 [online] February 1997 [retrieved on 1999-12-13]. Retrieved from the Internet via <URL: http://manuals.mchp.siemens.de/servers/bs2_man/man_us/utm/v4_0/ utm_kon.pdf>	7-10
A	BS2000/OSD-BC V1.0, Performance Handbook, Chapter 5.3.1: Managing the resource main memory, especially: Deactivation (p.248), Waiting Time runout Control (p.255),Paging management algorithms (pp.256-257), ID: U1794-J-Z125-6-7600 [online] February 1997 [retrieved on 1999-12-13]. Retrieved from the Internet via <URL: http://manuals.mchp.siemens.de/servers/bs2_man/man_us /bs2_bc/V1_0/perform.pdf>	7-10
A	WO 97/35262 A (HITACHI) 25 September 1997 (25.09.97), summary.	1-22

☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:
„A" document defining the general state of the art which is not considered to be of particular relevance
„E" earlier application or patent but published on or after the international filing date
„L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
„O" document referring to an oral disclosure, use, exhibition or other means
„P" document published prior to the international filing date but later than the priority date claimed
„T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
„X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
„Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
„&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
14 December 1999 (14.12.99)	14 January 2000 (14.01.00)
Name and mailing adress of the ISA/AT Austrian Patent Office Kohlmarkt 8-10; A-1014 Vienna Facsimile No. 1/53424/535	Authorized officer Fastenbauer Telephone No. 1/53424/447

Form PCT/ISA/210 (second sheet) (July 1998)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/SG 99/00009

Patent document cited in search report			Publication date	Patent family member(s)			Publication date
US	A	5603031	11-02-1997	AU	A1	72164/94	06-02-1995
				EP	A2	634719	18-01-1995
				EP	A3	634719	03-01-1996
				JP	A2	7182174	21-07-1995
				JP	T2	7509799	26-10-1995
				WO	A1	9502219	19-01-1995
WO	A1	9735262	25-09-1997			none	